IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Metaheuristic Algorithms for Multi-mode Multi-project Scheduling with the Objective of Positive Cash flow Balance

Yukang He[1], Jingwen Zhang[2], and Zhengwen He[3]

[1]School of Engineering & Applied Science, Aston University, Birmingham, B4 7ET, United Kingdom
[2]School of Management, Northwestern Polytechnical University, Xi'an 710072, China
[3]School of Management, Xi'an Jiaotong University, Xi'an 710049, China

Corresponding author: Zhengwen He (zhengwenhe@mail.xjtu.edu.cn).

The research was supported by the National Natural Science Foundation of China under Grants No. 71971173, 71572148, 71871176, and 71371150, and the Fundamental Research Funds for the Central Universities under Grants No. 3102019JC02.

ABSTRACT This paper investigates the problem of how to achieve a positive cash flow balance by multi-mode multi-project scheduling. First, based on formulating cash flows for the projects, we construct an optimization model in the multi-mode multi-project context that can minimize the maximum cash flow gap and, thus, balance cash flow positively by arranging optimal execution modes and start times for activities. Then, we prove the NP-hardness of the studied problem and design two metaheuristic algorithms, namely tabu search (TS) and simulated annealing (SA), which search the desirable solutions in nested and mixed ways, respectively. Finally, taking the multi-start iterative improvement (MSII) as comparison algorithm, the performance of the two algorithms developed is evaluated through a computational experiment performed on a data set generated randomly using ProGen. From the research results, the following conclusions are drawn: The TS and SA are more suitable for solving the smaller and larger problems, respectively, while the nested searching structure may enhance the algorithm's efficiency. With the increase of the advance payment proportion, number of milestone activities, client's payment proportion, or project deadline, the contractor's maximal cash flow gap decreases.

INDEX TERMS Project scheduling, Optimization model, Metaheuristic algorithm, Positive cash flow balance, Multi-mode multi-project context

I. INTRODUCTION

Cash flow management, which involves the forecasting, planning, monitoring, and controlling of the cash flow, is recognized as a critical issue in project management [1]. Over the course of a project, a series of cash flows occur in the following two forms: cash outflow—induced mainly by the execution of activities and the use of resources such as labour, equipment, and materials—and cash inflow—resulting generally from payments for the completion of specified parts of the project according to the contract terms. Based on this, it is easy to understand that during the implementation of the project, keeping a positive balance between cash outflow and inflow is key to effective cash flow management, since, if the outflow cannot be covered by the inflow in a timely manner, the contractor may not be able to implement the project smoothly. In such a case, the contractor has to raise money externally to cover the gap between cash outflow and inflow and hence incur an extra financing cost. Moreover, when the gap exceeds the contractor's financing capacity, the problem may cause project failure or even the bankruptcy of the contractor [2].

Essentially, the distribution of cash flow over the course of the project is closely related to the arrangement of the project schedule. On the one hand, the amount of cash outflow depends on the selection of each activity's execution mode, which represents the capital devoted to it and the time it consumes [3]; and when it occurs is mainly determined by the assigned start time for the activity. On the other hand, as the client often pays the contractor based on the progress of the project according to payment terms stipulated in the contract, each cash inflow (payment amount) and its timing rely primarily on the achievement of the project schedule. As a result, through reasonable project scheduling, which includes choosing an execution mode

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2944746, IEEE Access

and arranging a start time for each activity, the contractor can coordinate cash outflow and inflow effectively, thus, achieving a positive cash flow balance.

In fact, this problem is more meaningful in a multi-project context since, in practice, the contractor may implement multiple projects simultaneously [4]. In this context, the contractor must assign an execution mode and start time for activities in each project and manage all the cash flows for the different projects as a whole. During the implementation of a given project, if the positive balance between cash outflow and inflow falls short, to ensure the project continues smoothly, the contractor needs to extract cash from the other projects or raise money externally to cover the cash flow gap. When this happens, the implementation of the projects could be affected and the contractor's cost could increase. Thus, in the multi-project context, the contractor needs to arrange the project schedules carefully and try to achieve a positive cash flow balance for all projects during their implementation.

To address this, in this paper, we investigate a multi-mode multi-project scheduling problem to achieve a positive balance between cash outflow and inflow. The problem we look at is when a contractor is implementing multiple projects concurrently, with each project having its own deadline and activities being performed in one of several alternative modes. At the beginning of the projects, the client makes an advance payment to the contractor, hence generating a cash inflow for the contractor. Over the course of the projects, the cash outflows, namely the costs for performing activities, occur at the start of the corresponding activities while their amounts depend upon the selection of execution mode of the activities. In each project, some activities are defined as milestone activities by the client and at the completion of these activities, the cash inflows, namely the clients' progress payments, are received based on the accumulative earned value of the activities completed by the contractor. During the implementation of the projects, the cash flow can be shared among the projects freely and at a certain time, the cash flow gap is defined as the accumulative cash outflow minus the accumulative cash inflow. Our aim is to arrange the project schedule, which consists of the execution mode and start time of activities in all the projects, to minimize the cash flow gap so that the contractor achieves a positive balance between cash outflow and inflow. We believe that this research, which has not been performed previously to the best of our knowledge, provides valuable decision support for contractors to manage their cash flows.

The remainder of this paper is organized as follows. We present a literature review in Section II. Section III constructs the optimization model and analyzes the complexity of the problem. Section IV designs mateheuristic algorithms for the model while Section V provides a computational experiment. We present our conclusions in Section VI.

II. LITERATURE REVIEW

Project scheduling, a classical problem in project management, has gained increasing attention in the literature over the years [5]. In this area, a relevant literature stream is the capital-constrained max-NPV project-scheduling problem where the investment in project activities is constrained by capital constraints while payments are reinvested in the project to maximize the net present value of the cash flow [6]. In addressing this issue, Smith-Daniels and Smith-Daniels [7] treated materials and capital constraints in an integrated fashion to present an approach for the problem. Due to the intractability of this problem, Smith-Daniels et al. [8] presented three heuristic procedures and tested their performance in solving relatively large capital-constrained projects. Özdamar [9] invoked a multi-mode capital-constrained problem where the contractor had to construct and reconstruct schedules during the progress of the projects to maintain a positive cash balance dynamically. Liu and Wang [10] established a resource-constrained project scheduling model that maximized net cash flow to optimize project profit from the perspective of contractors. By combining the project payment-scheduling problem with the capital-constrained problem, He et al. [11] assessed a multi-mode capital-constrained project payment scheduling problem and developed metaheuristics to solve it.

Different from the researches aforementioned where the objective is to maximize the NPV of project, Elazouni and Gab-Allah [12] investigated how to produce financially feasible schedules that balanced the financing requirements of activities at any period with the cash available during that same period. The proposed problem was named the finance-based scheduling problem where the total project duration was minimized and the finance-availability constraint was fulfilled in the meantime. To account for large-size projects, Ali and Elazouni [13] and Alghazi et al. [14] applied genetic algorithms to develop finance-based schedule models while Elazouni et al. [15] and Al-Shihabi and AlDurgam [16] used simulated annealing and max-min ant system algorithms to deal with the problem, respectively. Further research efforts were made by Fathi and Afshar [17] and Elazouni and Abido [18] to consider multiple objectives in integrating a project's cash flow with its schedule, and by Liu and Wang [19] and El-Abbasy et al. [20] to take project finance into account in multi-project scheduling context. Under the objective of minimizing the maximal cash flow gap, He et al. [21] used variable neighbourhood search and tabu search to tackle a discrete time/cost trade-off problem for a single project.

In project scheduling, another relevant literature stream is the multi-project scheduling problem, which has been given more attention in recent years. For instance, Homberger [22] integrated a restart evolution strategy with a multi-agent system for solving the decentralized resource-constrained multi-project scheduling problem. Chen and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2944746, IEEE Access

Shahandashti [23] developed a hybrid of genetic algorithm and simulated annealing for generic multi-project scheduling problems with multiple resource constraints. Browning and Yassine [24] addressed the static resource-constrained multi-project scheduling problem with two lateness objectives and conducted a comprehensive analysis of 20 priority rules on 12,320 test problems. Using an auction-based negotiation approach for the resource intervals, Adhau et al. [25] presented a distributed multi-agent system to allocate multiple types of shared resources among multiple competing projects. Can and Ulusoy [26] considered a non-preemptive, zero-time lag, multi-project scheduling problem and used a two-stage decomposition approach to reformulate the problem as a hierarchy of 0-1 mathematical programming models. Supposing that each project had an assigned due date, activities could be performed in alternative modes, and resources did not have to be shared among projects, Beşikci et al. [27] investigated a multi-project scheduling problem and designed a two-phase and a monolithic genetic algorithm as two solution approaches for the problem.

From the brief literature review above, it can be seen clearly that the problem studied in this paper has a remarkable distinction from the existing researches on project scheduling. For this fact, we believe that the research may have an important implication for the research area.

III. PROBLEM FORMULATION

A. OPTIMIZATION MODEL

Let us assume that a contractor needs to implement H projects concurrently. Project h (h=1,2,…,H) is represented as an activity-on-node network wherein nodes denote the activities and arcs, the finish–start precedence constraints with a time lag of zero. In the project, there are n^h activities of which activities 1 and n^h are the dummy start and end activities, respectively, and the others are all non-dummy activities. The earned value of activity i is v_i and depending on the capital devoted, activity i (i=1,2,…,n^h) can be executed in one of E_i alternative modes. When activity i is assigned to be executed in mode m_i (m_i=1,2,…,E_i), its duration and cost are d_{im_i} and c_{im_i}, respectively. It should be noted that as the two dummy activities do not exist in reality, their d_{im_i}, c_{im_i}, and v_i, are a constant that equals 0. For project h, the deadline and contract price are D^h and U^h ($U^h = \sum_{i=1}^{n^h} v_i$), respectively.

The cash outflow for the contractor to complete activity i, namely, c_{im_i}, is assumed to occur at the activity's start time. Let us suppose that under the constraints of the project deadline D^h and a precedence relationship among the project's activities, the start time of activity i is arranged as s_i. Then, the execution mode and start time of all the activities, namely m_i and s_i, constitute a schedule for project h, represented as (M^h, S^h), where $M^h = (m_1, m_2, \ldots, m_{n^h})$ and $S^h = (s_1, s_2, \ldots, s_{n^h})$. Among the H projects, we represent the start time of the project that is begun earliest as $s_1^{\min}$ and the completion time of the project that is completed latest as $s_{n^h}^{\max}$. Given a (M^h, S^h), the contractor's accumulative cash outflow at time t ($t \in [s_1^{\min}, s_{n^h}^{\max}]$), which is denoted as ACO_t, is calculated using the formula $ACO_t = \sum_{h=1}^{H} \sum_{i=1}^{n^h} \sum_{s_i \le t} c_{im_i}$.

Let γ^h ($0 \le \gamma^h \le 1$) be the proportion of the advance payment of project h and at the beginning of the project, the client makes an advance payment, $\gamma^h \cdot U^h$, to the contractor. Among the n^h activities, there are K^h ($K^h \le n^h$) activities defined as milestone activities by the client. During the implementation of the project, when a milestone activity is finished, the client makes a progress payment to the contractor. Note that since the last payment is often arranged at the completion of the project, the dummy end activity n^h must be a milestone activity. The amount of the k-th progress payment, p_k (k=1,2,…,K^h–1), equals the product of the contractor's earned value accumulated from the (k–1)-th payment to this payment and the payment proportion of project h, θ^h ($0 \le \theta^h \le 1$). In p_k, the advance payment should be deducted according to the proportion, γ^h; as a result, p_k (k=1,2,…,K^h–1) is computed by the formula, $p_k = (\theta^h - \gamma^h) \cdot (\sum_{i \in AS_k^h} v_i - \sum_{i \in AS_{k-1}^h} v_i)$, where AS_k^h and AS_{k-1}^h are the sets of the activities that are finished by the k-th and (k–1)-th payments, respectively. When the project is finished, the sum of payments must equal the contract price of the project, hence, the last payment, p_{K^h}, is calculated by $p_{K^h} = U^h - (\gamma^h \cdot U^h + \sum_{k=1}^{K^h-1} p_k)$. We denote the accumulative cash inflow at time t($t \in [s_1^{\min}, s_{n^h}^{\max}]$) as ACI_t. Then, given a (M^h, S^h), ACI_t is computed using the formula, $ACI_t = \sum_{h=1}^{H} \sum_{s_1 \le t} (\gamma^h \cdot U^h) + \sum_{h=1}^{H} \sum_{k=1}^{K^h} \sum_{(s_{ik} + d_{ikm_k}) \le t} p_k$, where i_k represents the milestone activity to which the k-th payment is attached.

Using ACO_t and ACI_t, we define the cash flow gap at time t in the course of the H projects as $G_t = ACO_t - ACI_t$, and thus, the maximum cash flow gap, $G_{\max}$, is obtained using the formula $G_{\max} = \max_{t \in [s_1^{\min}, s_{n^h}^{\max}]} \{G_t\}$. Obviously, for all the H projects implemented concurrently, if the G_t<0 the contractor's cash flow is in a positive balance status at time t and if the $G_{\max}$<0 the contractor maintains a positive balance between cash outflow and inflow throughout the course of all H projects. Otherwise, the contractor has to raise money no less than the $G_{\max}$ with an additional financing cost to cover cash flow gaps so that the projects can be implemented smoothly. Based on the discussion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2944746, IEEE Access

above, we formulate the multi-mode multi-project scheduling problem with the objective of positive cash flow balance as the following non-linear integer programming model, where $G_{\max}$ is minimized by arranging (M^h, S^h) optimally.

$$\text{Min } G_{\max} = \max_{t\in[s_1^{\min}, s_{n^h}^{\max}]}\{G_t\} \quad (1)$$

$$\text{s.t. } s_i + d_{im_i} \le s_j,\ (i,j)\in A^h,\ h=1, 2, \ldots, H \quad (2)$$

$$s_{n^h} \le D^h,\ h=1, 2, \ldots, H \quad (3)$$

$$p_k = (\theta^h - \gamma^h)\cdot\Big(\sum_{i\in AS_k^h} v_i - \sum_{i\in AS_{k-1}^h} v_i\Big),\ k=1,2,\ldots,K^h-1,\ h=1,2,\ldots,H \quad (4)$$

$$p_{K^h} = U^h - (\gamma^h\cdot U^h + \sum_{k=1}^{K^h-1} p_k),\ h=1, 2, \ldots, H \quad (5)$$

$$ACO_t = \sum_{h=1}^{H}\sum_{i=1}^{n^h}\sum_{s_i\le t} c_{im_i},\ t\in[s_1^{\min}, s_{n^h}^{\max}] \quad (6)$$

$$ACI_t = \sum_{h=1}^{H}\sum_{s_1\le t}(\gamma^h\cdot U^h) + \sum_{h=1}^{H}\sum_{k=1}^{K^h}\sum_{(s_{ik}+d_{ik m_{ik}})\le t} p_k,\ t\in[s_1^{\min}, s_{n^h}^{\max}] \quad (7)$$

$$G_t = ACO_t - ACI_t,\ t\in[s_1^{\min}, s_{n^h}^{\max}] \quad (8)$$

m_i and s_i are a nonnegative integer, $i = 1, 2, \ldots, n^h$, and $h = 1, 2, \ldots, H$ (9)

In the model constructed above, the objective is to minimize the contractor's maximum cash flow gap in the multi-project context. Constraint (2) maintains the precedence feasibility, where A^h is the set of the precedence relationships among the activities. The deadline is imposed for project h by constraints (3). Constraints (4) and (5) are the formulae used to determine payments, whereas constraints (6) and (7) are used to calculate the accumulative cash outflow and inflow at time t, respectively. Constraint (8) is used to compute the cash flow gaps, and constraint (9) defines the value scope of the decision variables.

B. COMPLEXITY OF THE PROBLEM

Without any loss of generality, we let H=1, γ^1=1, and θ^1=0. This means that there is only one project for contractor to complete and in the project, the advance payment equals the contract price, U^1, and thus there are no progress payments occurring during the execution of the project. In such a case, the maximal cash flow gap, which must occur at the completion of the project, is equivalent to the total cost of the project, $\sum_{i=1}^{n^1} c_{im_i}$, minus the contract price, U^1. Since U^1 is a given constant, the studied problem is simplified to assigning modes and start times of activities to minimize the total cost of the single project under the deadline constraint, which in fact is the P_C|T in the discrete time/cost trade-off problem [28]. In other words, the problem studied in this paper can be regarded as a generalization of the P_C|T when considering progress payments in the multi-project context. As the P_C|T has been proven to be strongly NP-hard for general project networks [29], the studied problem must be strongly NP-hard as well.

IV. METAHEURISTIC ALGORITHMS

Due to the strong NP-hardness of the studied problem, we use two well-known metaheuristics, i.e. tabu search (TS) and simulated annealing (SA) which are originally developed by Glover [30] and Metropolis et al. [31] respectively and have been successfully applied to a number of project scheduling problems [32], for the solution of the problem. In this section, we describe the common features of the two metaheuristic algorithms first and then present the design of the TS and SA.

A. COMMON FEATURES

1) SOLUTION REPRESENTATION

A solution for the problem is represented by using the following two sets.

- Execution mode set: This set consists of all the M^hs for H projects and is represented as *EMS*, $EMS=(M^1,M^2,\ldots,M^H)$, which determines the arrangement of the execution modes of activities in the projects.
- Time deviation set: This set consists of H lists and is represented as *TDS*, $TDS=(TD^1,TD^2,\ldots,TD^H)$. In *TDS*, list TD^h (h=1,2,…,H) includes n^h elements and the i-th (i=1,2,…,n^h) element, which is denoted as Δ_i ($\Delta_i\in[0, ls_i-es_i]$ where ls_i and es_i are the latest and earliest start times of activity i respectively), indicates how many time units of activity i's start time deviate from es_i.

For a project, we let *EAS* represent the set of eligible activities, such as the unscheduled activities that have all predecessor activities scheduled. Then, a solution, (*EMS*, *TDS*), can be transformed into a schedule of the projects using the decoding procedure described as follows.

Step 1. Initialize the *EAS* and define the start time for each project, i.e., $EAS := \{1\}$ and input the value of s_1.

Step 2. According to the *EMS* and *TDS*, determine the d_{im_i} and Δ_i of activities in each project, respectively.

Step 3. For each project, update the *EAS* by removing the scheduled activities from the *EAS* and adding the new eligible activities to it. Judge whether the *EAS* is empty. If the answer is yes, go to step 5; otherwise, go to step 4.

Step 4. For each project, on the basis of d_{im_i} and the precedence relationship between activities, arrange each activity in the *EAS* to start as early as possible,

thereby obtaining an s_i. Then, $s_i := s_i + \Delta_i$ and go to step 3.

Step 5. Terminate the procedure and all the s_is obtained for a project, h, constitute an S^h. The generated S^h and the M^h in the EMS form a schedule, (M^h, S^h) for project h. Output the (M^h, S^h) for each project.

2) STARTING SOLUTION

The starting solution, denoted as (EMS^{star}, TDS^{star}), is generated according to the following steps.

Step 1. For all the activities in each project, arrange each execution mode as one with the lowest cost. Check whether the critical path length of the project network is greater than the project deadline. If the answer is no, we receive a feasible M^h that does not violate the project deadline constraint and incurs the lowest total cost. Otherwise, we select an activity on the critical path and change the execution mode to shorten the critical path length with the minimal cost increment. We repeat this operation until the critical path length is less than or equivalent to the project deadline, thereby obtaining the feasible M^h. All the generated M^hs constitute an EMS^{star}.

Step 2. For each project, determine the duration of activities according to the M^h given by the EMS^{star}. Without violating the constraints of precedence and project deadline, arrange the milestone activities to start as soon as possible while the non-milestone activities to start as late as possible, thereby generating an S^h. Based on the generated S^h, determine TD^h for each project and all the obtained TD^hs compose a TDS^{star}.

Step 3. Output the starting solution, i.e. the (EMS^{star}, TDS^{star}) obtained finally.

3) NEIGHBOUR GENERATION

The current solution is represented as (EMS^{curr}, TDS^{curr}). Based on EMS^{curr} and TDS^{curr}, the neighbour EMS and TDS, which are denoted as EMS^{neig} and TDS^{neig} respectively, are generated using the following two operators.

- Mode change (MC): Select a random M^h from the EMS^{curr}. In the selected M^h, choose an m_i arbitrarily and under the constraint of the project deadline, change its value to another available one randomly. This makes the EMS^{curr} become a new one and we denote it as EMS^{neig}, where other M^hs remain unchanged.
- Deviation change (DC): Select a TD^h from the TDS^{curr} in a random fashion. Arbitrarily choose a Δ_i in the selected TD^h and under the constraint of the project deadline, change its value to another one within [0, ls_i–es_i] randomly. This makes the TDS^{curr} become a new one represented as TDS^{neig}, in which other TD^hs remain unchanged.

B. TABU SEARCH

1) MOVES

Corresponding to the two neighbour generation operators, the moves in the TS are defined as follows:

- Move for MC: It is a quadruple of (position number of the selected M^h in the EMS^{curr}, position number of the chosen m_i in the selected M^h, original value, new value). E.g. if in the EMS^{curr}, m_5 of M^2 is selected and its value is changed from 1 to 2, then the move is expressed as (2,5,1,2), implying that the mode of activity 5 in project 2 is changed from 1 to 2. Consequently, the reverse move, which has the form of (2,5,1), is added to the tabu list, preventing the mode of this activity from being changed back to 1.
- Move for DC: It is a quadruple of (position number of the selected TD^h in the TDS^{curr}, position number of the chosen Δ_i in the selected TD^h, original value, new value). E.g. if Δ_3 of TD^1 is changed from 8 to 6 in the TDS^{curr}, then the move is expressed as (1,3,8,6), indicating that the time deviation of activity 3 of project 1 is changed from 8 to 6. In consequence, the reverse move is denoted as (1,3,8) and it is added to the tabu list, forbidding the time deviation of activity 3 being assigned as 8 once again.

2) TABU LIST AND STOP CRITERION

In the TS, there are two tabu lists, TL^{MC} and TL^{DC}, which are used to store the tabu moves for MC and TC, respectively. The two tabu lists are managed according to the First-in-First-out rule and all the moves on the tabu lists are forbidden. However, if a tabu move can generate a solution better than the best found so far, its tabu status may be canceled so that the algorithm can move to this solution. We take a given number of the feasible solutions visited, Num^{stop}, as the stop criterion of the TS. In other words, when the number of feasible solutions explored by the TS reaches Num^{stop} it terminates and outputs the best solution saved as the desirable one.

3) IMPLEMENTATION STEPS

Considering that when scheduling the projects, the start time of the activities has to be arranged after their execution modes have been determined, we design the TS as the following two nested loops: The inner loop seeks the desirable TDS under the given EMS while the outer loop searches the desirable EMS based on the results returned by the inner loop. We denote the neighbour solution as (EMS^{neig}, TDS^{neig}), the best solution found during the searching process as (EMS^{best}, TDS^{best}), the G_{max} under the starting, current, neighbour, and best solutions as G_{max}^{star}, G_{max}^{curr}, G_{max}^{neig}, and G_{max}^{best} respectively, and the number of the feasible solutions visited during the searching process as Num. Then, the implementation steps of the TS are described as follows, where INNERLOOP(EMS^{neig}) represents the inner loop, which finds the desirable TDS and the corresponding G_{max} under the given EMS^{neig}. Note that in INNERLOOP(EMS^{neig}), the stop criterion is defined as $Num^{stop\text{-}in}$, which is a number of the feasible TDSs required to be visited in the inner loop.

Step 1. Initialize TL^{MC}, define Num^{stop}, and Num :=0.

Step 2. Generate a starting solution, (EMS^{star}, TDS^{star}). Compute the G_{max} under the (EMS^{star}, TDS^{star}) and denote it as G_{max}^{star}. (EMS^{curr}, TDS^{curr}) :=(EMS^{star}, TDS^{star}), $G_{max}^{curr} := G_{max}^{star}$, ($EMS^{best}$, TDS^{best}) :=(EMS^{star}, TDS^{star}), $G_{max}^{best} := G_{max}^{star}$, and Num :=Num+1.

Step 3. Based on the EMS^{curr}, generate an EMS^{neig} by operator MC. Invoke INNERLOOP(EMS^{neig}) and take the desirable TDS and G_{max} returned as TDS^{neig} and G_{max}^{neig}, respectively.

Step 4. Assess whether the move from the EMS^{curr} to the EMS^{neig} is in the TL^{MC}. If the answer is yes, go to step 5; otherwise, go to step 6.

Step 5. Assess whether the G_{max}^{neig} is less than the G_{max}^{best}. If the answer is yes, (EMS^{curr}, TDS^{curr}) :=(EMS^{neig}, TDS^{neig}), $G_{max}^{curr} := G_{max}^{neig}$, ($EMS^{best}$, TDS^{best}) :=(EMS^{neig}, TDS^{neig}), $G_{max}^{best} := G_{max}^{neig}$, update TL^{MC}, and go to step 7; otherwise, go to step 7 directly.

Step 6. (EMS^{curr}, TDS^{curr}) :=(EMS^{neig}, TDS^{neig}) and $G_{max}^{curr} := G_{max}^{neig}$, and if the G_{max}^{neig} is less than the G_{max}^{best}, (EMS^{best}, TDS^{best}) :=(EMS^{neig}, TDS^{neig}) and $G_{max}^{best} := G_{max}^{neig}$. Update the TL^{MC} and go to step 7.

Step 7. Num :=Num+$Num^{stop\text{-}in}$. Assess whether $Num \geq Num^{stop}$. If the answer is yes, go to step 8; otherwise, go to step 3.

Step 8. Output the desirable results, i.e. the (EMS^{best}, TDS^{best}) and G_{max}^{best} obtained finally.

In INNERLOOP(EMS^{neig}), we denote the number of the feasible TDSs visited during the searching process as Num^{in}. Then, the implementation steps of the inner loop are as follows, where $TDS^{star\text{-}in}$, $TDS^{curr\text{-}in}$, $TDS^{neig\text{-}in}$, and $TDS^{best\text{-}in}$ are the starting, current, neighbour, and best TDSs respectively, while $G_{max}^{star\text{-}in}$, $G_{max}^{curr\text{-}in}$, $G_{max}^{neig\text{-}in}$, and $G_{max}^{best\text{-}in}$ are their G_{max}s respectively.

INNERLOOP(EMS^{neig})

Step 1. Input EMS^{neig}, initialize TL^{DC}, define $Num^{stop\text{-}in}$, and Num^{in} :=0.

Step 2. Under the EMS^{neig}, generate a starting TDS, $TDS^{star\text{-}in}$. Compute the G_{max} under the (EMS^{neig}, $TDS^{star\text{-}in}$) and denote it as $G_{max}^{star\text{-}in}$. $TDS^{curr\text{-}in}$:= $TDS^{star\text{-}in}$, $G_{max}^{curr\text{-}in} := G_{max}^{star\text{-}in}$, $TDS^{best\text{-}in}$:=$TDS^{star\text{-}in}$, $G_{max}^{best\text{-}in} := G_{max}^{star\text{-}in}$, and Num^{in} :=Num^{in}+1.

Step 3. Based on the $TDS^{curr\text{-}in}$, generate a neighbour TDS, $TDS^{neig\text{-}in}$, by operator DC. Compute the G_{max} under the (EMS^{neig}, $TDS^{neig\text{-}in}$) and denote it as $G_{max}^{neig\text{-}in}$.

Step 4. Assess whether the move from the $TDS^{curr\text{-}in}$ to the $TDS^{neig\text{-}in}$ is in the TL^{DC}. If the answer is yes, go to step 5; otherwise, go to step 6.

Step 5. Assess whether the $G_{max}^{neig\text{-}in}$ is less than the $G_{max}^{best\text{-}in}$. If the answer is yes, $TDS^{curr\text{-}in}$:=$TDS^{neig\text{-}in}$, $G_{max}^{curr\text{-}in} := G_{max}^{neig\text{-}in}$, $TDS^{best\text{-}in}$:=$TDS^{neig\text{-}in}$, $G_{max}^{best\text{-}in} := G_{max}^{neig\text{-}in}$, update the TL^{DC}, and go to step 7; otherwise, go to step 7 directly.

Step 6. $TDS^{curr\text{-}in}$:=$TDS^{neig\text{-}in}$ and $G_{max}^{curr\text{-}in} := G_{max}^{neig\text{-}in}$, and if the $G_{max}^{neig\text{-}in}$ is less than the $G_{max}^{best\text{-}in}$, $TDS^{best\text{-}in}$:= $TDS^{neig\text{-}in}$ and $G_{max}^{best\text{-}in} := G_{max}^{neig\text{-}in}$. Update the TL^{DC} and go to step 7.

Step 7. Num^{in} :=Num^{in}+1. Assess whether the Num^{in} reaches the Num^{stop}. If the answer is yes, go to step 8; otherwise, go to step 3.

Step 8. Return the desirable TDS and G_{max}, i.e., the $TDS^{best\text{-}in}$ and $G_{max}^{best\text{-}in}$ obtained finally.

C. SIMULATED ANNEALING

1) COOLING SCHEME

The SA is specified by the cooling scheme, which is constituted of the initial temperature, the cooling rate, the Markov chain length, and the stop criterion. The initial temperature, $Temp^{init}$, is calculated by $Temp^{init}$=($G_{max}^{star} - G_{max}^{max}$)/ln$Prob^{init}$, where G_{max}^{max} is the maximal G_{max} among the 50 neighbour solutions of the starting solution, while $Prob^{init}$, which is set at 0.9 in this application, the initial acceptance ratio defined as the number of accepted neighbour solutions divided by that of the proposed neighbour solutions. Beginning from $Temp^{init}$, the temperature, $Temp$, is progressively reduced according to a certain cooling rate, CR, and under a given $Temp$, the number of transitions is determined by the Markov chain length, MCL. Similar with the TS, the stop criterion of the SA is defined as a given number of the feasible solutions visited, Num^{stop}, as well.

2) IMPLEMENTATION STEPS

Different from the TS, the SA searches the desirable solution, which consists of the desirable EMS and TDS, in a mixed way. In other words, when generating a neighbour solution during the searching process of the SA, the two operators are chosen in a random way. The implementation steps of the SA are as follows, where Num has the same meaning with that in the TS while $TNum$ denotes the number of the feasible solutions visited under a given temperature.

Step 1. Define CR, MCL, and Num^{stop}, determine $Temp^{init}$, $Temp$:=$Temp^{init}$, $TNum$:=0, and Num :=0.

Step 2. Generate a starting solution, (EMS^{star}, TDS^{star}). Compute the G_{max} under the (EMS^{star}, TDS^{star}) and denote it as G_{max}^{star}. (EMS^{curr}, TDS^{curr}) :=(EMS^{star}, TDS^{star}), $G_{max}^{curr} := G_{max}^{star}$, and Num :=Num+1.

Step 3. Select an operator from MC and DC according to an equal probability to generate an EMS^{neig} (or a TDS^{neig}). The generated EMS^{neig} (or TDS^{neig}) and

TDS^{curr} (or EMS^{curr}) form a new solution, (EMS^{neig}, TDS^{curr}) (or (EMS^{curr}, TDS^{neig})), and we take this new solution as (EMS^{neig}, TDS^{neig}). Compute the G_{max} under the (EMS^{neig}, TDS^{neig}) and denote it as G_{max}^{neig}.

Step 4. $\Delta G_{max} := G_{max}^{curr} - G_{max}^{neig}$ and then assess whether ΔG_{max} is greater than 0. If the answer is yes, (EMS^{curr}, TDS^{curr}) :=(EMS^{neig}, TDS^{neig}), $G_{max}^{curr} := G_{max}^{neig}$, and go to step 6; otherwise, go to step 5.

Step 5. Generate a random number from U[0, 1]. If this number is not greater than $e^{\Delta G_{max}/Temp}$, (EMS^{curr}, TDS^{curr}) :=(EMS^{neig}, TDS^{neig}), $G_{max}^{curr} := G_{max}^{neig}$, and go to step 6; otherwise, go to step 6 directly.

Step 6. Num :=Num+1. Assess whether $Num \geq Num^{stop}$. If the answer is yes, go to step 9; otherwise, go to step 7.

Step 7. $TNum$:=$TNum$+1. Assess whether $TNum \geq MCL$. If the answer is yes, go to step 8; otherwise, go to step 3.

Step 8. $Temp$:=$Temp \cdot CR$, $TNum$:=0, and go to step 3.

Step 9. Output the desirable results, i.e. the (EMS^{curr}, TDS^{curr}) and G_{max}^{curr} obtained finally.

V. COMPUTATIONAL EXPERIMENT

A. EXPERIMENTAL DESIGN

To evaluate the performance of the TS and SA developed in this paper, we utilize another method, multi-start iterative improvement (MSII) [33,34], to provide comparable computational efforts. In the implementation, the MSII starts from the same starting solution and employs the same neighbour generation mechanism as those used in the SA. During the search process, it chooses the most improving neighbour solution as the move and if there are no improving moves, restarts with another feasible solution generated randomly. The MSII stops and takes the best solution found as the desirable one when the number of the feasible solutions it has visited attains a certain value, Num^{stop}.

The experiment is conducted on a data set generated by ProGen project generator [35] using the parameter settings presented in Table I. The data set consists of 40 instances where the contractor needs to implement two projects concurrently and in each project, the number of non-dummy activities is set at 10, 20, 30, or 40. In the instances, the values of the key parameters, including D^h, θ^h, γ^h, and K^h, are set at three levels and a full factorial experiment of the four parameters with three levels results in 3^4=81 replicates for each instance and 40·81=3,240 ones as whole.

TABLE I
PARAMETER SETTINGS USED TO GENERATE THE DATA SET

Parameter	Setting
Number of projects, H	2
Number of non-dummy activities in projects, n^h–2	10, 20, 30, 40
Number of instances generated under a given number of non-dummy activities	10
Number of initial and terminal activities in projects	Randomly selected from 2, 3, and 4
Maximal number of successors and predecessors in projects	4
Number of execution modes for activities	2
Duration of activities with mode 1, d_{i1}	Randomly selected from U[1, 10]
Cost of activities with mode 1, c_{i1}	Randomly selected from U[1, 10]
Duration of activities with mode 2, d_{i2}	$\rho_1 \cdot d_{i1}$, where ρ_1, which is a special parameter defined for generating d_{i2}, is randomly selected from U[0.75, 0.95]
Cost of activities with mode 2, c_{i2}	$\rho_2 \cdot c_{i1}$, where ρ_2, which is a special parameter defined for generating c_{i2}, is randomly selected from U[1.05, 1.25]
Earned value of activities, v_i	$\rho_3 \cdot c_{i2}$, where ρ_3, which is a special parameter defined for generating v_i, is randomly selected from U[1.3, 1.5]
Advance payment proportion of projects, γ^h	0.04, 0.06, 0.08
Payment number of projects, K^h	4, 5, 6
Milestone activities	The dummy end activity is a milestone activity while other milestone activities are selected from all the non-dummy activities randomly
Payment proportion of projects, θ^h	0.7, 0.8, 0.9
Deadline of projects, D^h	$\rho_4 \cdot (CPL_{max}–CPL_{min})+CPL_{min}$, where CPL_{max} and CPL_{min} are the critical path length of networks when all the activities are executed with modes 1 and 2 respectively, and ρ_4, which is a special parameter defined for generating D, is set at 0.4, 0.6, and 0.8

We define the following four indices to evaluate the performance of the algorithms.

- *NBS*: The number of instances for which the algorithms find a solution equal to the best solution known.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2944746, IEEE Access

- *ARD*: Average relative per cent below the best solution known.
- *MRD*: Maximal relative per cent below the best solution known.
- *ACT*: Average computational time of the algorithm.
- *MCT*: Maximal computational time of the algorithm.

In the definitions of *NBS*, *ARD*, and *MRD*, the best solution known is defined as the solution whose G_{max} is the lowest among those found by the three algorithms, namely the TS, SA, and MSII. All the algorithms are coded and compiled using Microsoft Visual C++, and the computational experiment is performed on an Intel Core-based personal computer with a 2.60-GHz clock-pulse and 3.88-GB RAM. Based on a preliminary empirical test, the parameters of the algorithms are set as follows: $Num^{stop}=10000\cdot n^h$, $Num^{stop\text{-}in}=10\cdot n^h$, $CR=0.9$, $MCL=100\cdot n^h$, and the length of both TL^{MC} and TL^{DC} is set at 5.

B. PERFORMANCE COMPARISON OF THE ALGORITHMS

The computational results of the three algorithms are given in Table II. From the table, it can be seen that the SA and TS outperform the MSII clearly and with the increase of the number of activities, their superiorities augment. This result is not surprising because the intelligent search processes generally get an advantage over simple search procedures like the MSII and this advantage grows when the problem becomes more complex. Second, Table 2 also shows that the performance of the TS is a little better than that of the SA overall. This result may come from the following fact: Based on the characteristic of the studied problem, the TS uses the inner and outer loops to search the desirable *TDS* and *EMS*, respectively. However, in the SA, the desirable *EMS* and *TDS* are sought in a mixed way, where the operators MC and DC are selected randomly to generate neighbour solutions without any consideration of the problem's characteristic. This may lead to the searching structure of the TS is more organized and thus more reasonable than that of the SA, making the desirable solutions found by the TS are better than those obtained by the SA as whole. Finally, a further comparison of the TS and SA shows that as n^h increases, the results of the TS get worse while the reverse is true for the SA. This may be because the SA is more suitable for the solution of the larger problem due to its random nature, whereas the TS may have an advantage in tackling the relatively small problem. Therefore, as the scale of the problem rises, the SA tends to be more efficient than the TS.

TABLE II
PERFORMANCE COMPARISON OF THE ALGORITHMS

	TS					SA					MSII				
n^h	*NBS*	*ARD* (%)	*MRD* (%)	*ACT* (s)	*MCT* (s)	*NBS*	*ARD* (%)	*MRD* (%)	*ACT* (s)	*MCT* (s)	*NBS*	*ARD* (%)	*MRD* (%)	*ACT* (s)	*MCT* (s)
12	614	1.13	2.50	15.75	19.95	486	2.95	3.78	16.65	18.78	256	5.31	9.95	16.44	19.32
22	577	1.55	3.40	28.86	40.28	514	2.28	4.04	31.42	42.66	211	8.74	13.3	29.64	40.73
32	526	2.18	4.05	47.67	61.10	538	2.00	4.23	51.14	68.00	112	16.17	24.06	48.26	65.33
42	467	2.86	4.76	78.18	95.73	555	1.58	3.77	85.28	105.47	74	23.65	35.27	80.18	99.36
All instances	2,184	1.93	4.76	42.62	95.73	2,093	2.20	4.23	46.12	105.47	653	13.47	35.27	43.63	99.36

The computational times, which are indicated by the *ACT* and *MCT* indices, are as follows: The TS is the fastest, then the MSII, and the SA is the slowest. This phenomenon is explained below. Recall that in the TS, the nested loops are used to search for the desirable solution and when generating neighbour *TDS*s, the *EMS* remains unchanged. However, in the SA and MSII, the desirable *EMS* and *TDS* are sought in a mixed fashion, making the case where the *EMS* is changed while the *TDS* keeps unchanged may occur. Since the interval of $[0, ls_i–es_i]$ varies with the change of the *EMS*, the occurrence of this case may increases the probability of the generated neighbour solution being time infeasible. Hence, during the searching process of the SA and MSII, the algorithms may encounter more infeasible neighbour solutions than the TS. As a result, under the stop criterion that the algorithms have to visit the same number of the feasible solutions, the SA and MSII run a longer time than the TS although the TS needs to spend additional time to manage its tabu lists. Ultimately, with respect to the SA and MSII which start from the same starting solution and employ the same neighbour generation mechanism, it is easy to understand that the SA runs more slowly than the MSII because the former owns a more complicated searching structure than the latter.

C. EFFECTS OF THE KEY PARAMETERS ON THE MAXIMAL CASH FLOW GAP

The average value of the objective function for various cases, where the key parameters are set at different values, are presented in Table III. As shown, the maximal cash flow gap, G_{max}, decreases with the increase in γ^h, K^h, θ^h, or D^h. The reasons for these results are described as follows. First, as γ^h increases, the advance payment at the begining of the projects increases accordingly. This enhances the contractor's capital availability directly and hence makes G_{max} decrease. Second, when K^h goes up, the number of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2944746, IEEE Access

progress payments increases correspondingly and as a result, the average span between the two adjacent payments shortens. This causes the contractor's expense to be compensated more quickly, thus leading to G_{max} to go down. Third, because in the studied problem, the total payment must equal the contract price of the projects, the increase of θ^h can make a part of the last payment shift to the progress payments. The advancement of this part of the last payment can enhance the amount of the cash inflows during the execution of the projects and cause G_{max} to decrease. Finally, increasing the deadline of the projects D^h may relaxes the deadline constraint to some extent. Therefore, the contractor can save some costs for crashing activities and reduce the cash outflows subsequently, hence resulting in a decrease in G_{max}.

TABLE III
EFFECTS OF KEY PARAMETERS ON THE MAXIMAL CASH FLOW GAP

Parameters	Values	G_{max}		
		TS	SA	MSII
γ^h	0.04	37.60	37.63	40.48
	0.06	28.52	28.65	32.16
	0.08	20.43	20.48	23.72
K^h	4	32.78	32.91	35.31
	5	28.72	28.80	32.78
	6	25.07	25.05	28.26
θ^h	0.7	38.00	38.13	41.66
	0.8	28.48	28.53	31.75
	0.9	20.07	20.11	22.98
D^h	$0.4\cdot(CPL_{max}-CPL_{min})+CPL_{min}$	31.20	31.25	34.58
	$0.6\cdot(CPL_{max}-CPL_{min})+CPL_{min}$	28.88	28.91	32.05
	$0.8\cdot(CPL_{max}-CPL_{min})+CPL_{min}$	26.46	26.61	29.76

VI. CONCLUSIONS

In this paper, we propose a multi-mode multi-project scheduling problem with the objective to achieve a positive balance between cash outflow and inflow, where the contractor needs to implement multiple projects concurrently, with each project having its own deadline and activities being performed in one of several alternative modes. First, based on formulating cash outflow and cash inflow for the projects, we construct an optimization model in the multi-mode multi-project context, which can minimize the maximum cash flow gap and, thus, achieve the best positive cash flow balance by optimally arranging execution modes and start times of the activities. Then, we prove the NP-hardness of the studied problem and in the light of the characteristic of the constructed model, we design two metaheuristic algorithms, namely the TS and SA, which search the desirable solutions in the nested and mixed ways, respectively. Finally, taking the multi-start iterative improvement as comparison algorithm, the performance of the developed algorithms is evaluated through a computational experiment performed on a data set generated randomly using ProGen. In addition, based on the obtained computational results, the effects of several key parameters on the objective function are analyzed.

From the research results, the following conclusions are drawn: Compared with the MSII, the developed two metaheuristic algorithms can solve the studied problem more efficiently. The TS and SA are more suitable for obtaining a desirable solution of the smaller and larger problems, respectively, and the nested searching structure can improve the efficiency of the algorithms. As the advance payment proportion, number of milestone activities, client's payment proportion, or deadline of projects increase, the contractor's maximal cash flow gap decreases.

The research in this paper may provide decision support for the contractor to smooth their cash flows during the execution of multiple projects. However, it should be pointed out that in the paper, how to cover cash flow gaps economically is not considered and when scheduling projects, resource constraints are neglected as well. Therefore, in the future, we will extend the research to the above two cases that are more in line with reality.

REFERENCES

[1] A. Jiang, R. R. A Issa, and M. Malek, "Construction project cash flow planning using the Pareto optimality efficiency network model," *J. Constr. Eng. Manag.*, Vol. 17, no. 4, pp. 510–519, 2011.

[2] M. Ning, Z. He, N. Wang, and R. Liu, "Metaheuristics for project scheduling with the objective of minimizing contractor's cash flow gap under random activity durations," *IEEE Access*, Vol. 6, pp. 30547−30558, 2018.

[3] J. Węglarz, J. Józefowska, M. Mika, and G. Waligóra, "Project scheduling with finite or infinite number of activity processing modes — a survey," *Eur. J. Oper. Res.*, Vol. 208, no. 3, pp. 177–205, 2011.

[4] M. Engwall and A. Jerbrant, "The resource allocation syndrome: The prime challenge of multi-project management?" *Int. J. Proj. Manag.*, Vol. 21, pp. 403–409, 2003.

[5] C. Schwindt and J. Zimmermann, *Handbook of project management and scheduling*. Springer International Publishing AG, Berlin, 2014.

[6] W. S. Herroelen, P. Dommelen, and E. L. Demeulemeester, "Project network models with discounted cash flows: A guided tour through recent developments," *Eur. J. Oper. Res.*, Vol. 100, no. 1, pp. 97–121, 1997.

[7] D. E. Smith-Daniels and V. L. Smith-Daniels, "Maximizing the net present value of a project subject to materials and capital constraints," *J. Oper. Manage.*, Vol. 7, pp. 33–45, 1987.

[8] D. E. Smith-Daniels, R. Padman, and V. L. Smith-Daniels, "Heuristic scheduling of capital constrained projects," *J. Oper. Manage.*, Vol. 14, no. 3, pp. 241–254, 1996.

[9] L. Özdamar, "On scheduling project activities with variable expenditure rates," *IIE Trans.*, Vol. 30, no. 8, pp. 695–704, 1998.

[10] S. Liu and C. Wang, "Resource-constrained construction project scheduling model for profit maximization considering cash flow," *Autom. Constr.*, Vol. 17, pp. 966–974, 2008.

[11] Z. He, R. Liu, and T. Jia, "Metaheuristics for multi-mode capital-constrained project payment scheduling," *Eur. J. Oper. Res.*, Vol. 223, no. 3, pp. 605−613, 2012.

[12] A. M. Elazouni and A. A. Gab-Allah, "Finance-based scheduling of construction projects using integer programming," *J. Constr. Eng. Manag.*, Vol. 130, no. 1, pp. 15–24, 2004.

[13] M. M. Ali and A. Elazouni, "Finance-based CPM/LOB scheduling of projects with repetitive non-serial activities," *Constr. Manag. Econ.*, Vol. 27, no. 9, pp. 839–856, 2009.

[14] A. Alghazi, A. Elazouni, and S. Selim, "Improved genetic algorithm for finance-based scheduling," *J. Comput. Civ. Eng.*, Vol. 27, no. 4, pp. 379–394, 2013.

[15] A. Elazouni, A. Alghazi, and S. Selim, "Finance-based scheduling using meta-heuristics: discrete versus continuous optimization problems," *J. Financ. Manag. Prop. Constr.*, Vol. 20, no. 1, pp. 85–104, 2015.

[16] S. T. Al-Shihabi and M. M. AlDurgam, "A max-min ant system for the finance-based scheduling problem," *Comput. Ind. Eng.*, Vol. 110, pp. 264–276, 2017.

[17] H. Fathi and A. Afshar, "GA-based multi-objective optimization of finance-based construction project scheduling," *KSCE J. Civ. Eng.*, Vol. 14, no. 5, pp. 627–638, 2010.

[18] A. Elazouni and M. Abido, "Multiobjective evolutionary finance-based scheduling: Individual projects within a portfolio," *Autom. Constr.*, Vol. 20, no. 7, pp. 755–766, 2011.

[19] S. S. Liu and C. J. Wang, "Profit optimization for multiproject scheduling problems considering cash flow," *J. Constr. Eng. Manag.*, Vol. 136, no. 12, pp. 1268–1278, 2010.

[20] M. S. El-Abbasy, A. Elazouni, and T. Z. F.ASCE, "Generic scheduling optimization model for multiple construction projects," *J. Comput. Civ. Eng.*, Vol. 31, no. 4, pp. 04017003, 2017.

[21] Z. He, H. He, R. Liu, and N. Wang, "Variable neighbourhood search and tabu search for a discrete time/cost trade-off problem to minimize the maximal cash flow gap," *Comput. Oper. Res.*, Vol. 78, pp. 564−577, 2017.

[22] J. Homberger, "A multi-agent system for the decentralized resource-constrained multi-project scheduling problem," *Int. T. Oper. Res.*, Vol. 14, pp. 565–589, 2007.

[23] P. Chen and S. M. Shahandashti, "Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints," *Autom. Constr.*, Vol. 18, pp. 434–443, 2009.

[24] T. R. Browning and A. A. Yassine, "Resource-constrained multi-project scheduling: Priority rule performance revised," *Int. J. Prod. Econ.*, Vol. 126, pp. 212–228, 2010.

[25] S. Adhau, M. L. Mittal, and A. Mittal, "A multi-agent system for decentralized multi-project scheduling with resource transfers," *Int. J. Prod. Econ.*, Vol. 146, pp. 646–661, 2013.

[26] A. Can and G. Ulusoy, "Multi-project scheduling with two-stage decomposition," *Ann. Oper. Res.*, Vol. 217, pp. 95–116, 2014.

[27] U. Beşikci, U. Bilge, and G. Ulusoy, "Multi-mode resource constrained multi-project scheduling and resource portfolio problem," *Eur. J. Oper. Res.*, Vol. 240, pp. 22–31, 2015.

[28] P. De, E. J. Dunne, J. B. Ghosh, and C. E. Wells, "The discrete time-cost tradeoff problem revisited," *Eur. J. Oper. Res.*, Vol. 81, no. 2, pp. 225–238, 1995.

[29] P. De, E. J. Dunne, J. B. Ghosh, and C. E. Wells, "Complexity of the discrete time-cost tradeoff problem for project networks," *Oper. Res.*, Vol. 45, no. 2, pp. 302–306, 1997.

[30] F. Glover, "Future path for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, Vol. 13, no. 5, pp. 533–549, 1986.

[31] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, Vol. 21, no. 6, pp. 1087–1092, 1953.

[32] R. Pellerin, N. Perrier, and F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, https://doi.org/10.1016/j.ejor.2019.01.063, in press, 2019.

[33] M. Mika, G. Waligóra, and J. Węglarz, "Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times," *Eur. J. Oper. Res.*, Vol. 187, no. 3, pp. 1238–1250, 2008.

[34] G. Waligóra, "Discrete – continuous project scheduling with discounted cash flows—A tabu search approach," *Comput. Oper. Res.*, Vol. 35, pp. 2141–2153, 2008.

[35] R. Kolisch and A. Sprecher, "PSPLIB – A project scheduling problem library," *Eur. J. Oper. Res.*, Vol. 96, no. 1, pp. 205–216, 1996.